Math 260A — Mathematical Logic — Scribe Notes
UCSD — Spring Quarter 2012
Instructor: Sam Buss

Notes by: Sam Buss
May 11, 2012 — Part II

The Graph of Exponentiation — Simplifed Δ_0 Definition

These notes present a new, simplified Δ_0 definition of the graph of the base 2 exponentiation function $y = 2^x$. This definition is a key part of the bootstrapping process for the theory $I\Delta_0$, and is a key part of the approach we use for giving Δ_0 definitions for sequence coding functions.

The goal is to present Δ_0 definitions of graph of $x = 2^y$ along with the following functions:

$$
\begin{aligned}
|x| &= \lceil \log_2(x+1) \rceil, \quad \text{the length of } x\text{'s binary represenation.} \\
2^x = y &: \quad \text{the graph of the base 2 exponentiation function.} \\
\mathrm{Bit}(i,x) &= x/2^i \bmod 2, \quad \text{the } i\text{-th bit of } x. \\
\mathrm{Numones}(x) &= \text{the number of 1's in the binary representation of } x. \\
\beta(i,w) &= \text{the } i\text{-th member of the sequence coded by } w. \\
\mathrm{Len}(w) &= \text{the number of member in the sequence coded by } w.
\end{aligned}
$$

The material replaces the material on pages 91-94 of Chapter II of the *Handbook of Proof Theory* [1]. We use the notation and material developed in the earlier part of that chapter (and in the Math 260 class lectures), and presume this notation is familiar to the reader.

The original Δ_0 definition of the graph of exponentiation, formalized in the theory $I\Delta_0$ is due to Gaifman and Dimitracopoulos [2]. Proofs are also given in Hájek-Pudlák [3].

Compressed representation for consecutive counting. The first step in our bootstrapping of Δ_0 definitions is to define a *compressed* representation of consecutive counting is to encode the sequence of integers $1, 2, 3, 4, \ldots, k$ with a single value $w \in \mathbb{N}$ such $|w| = O(k)$. We wish to do this in such a way that we can pick out the location of each value i in the sequence in a natural way.

Let $\ell \in \mathbb{N}$. Define ℓ^* to be the maximum power of 2 that divides ℓ. That is

$$y = \ell^* \;\Leftrightarrow\; i^* \text{ is the largest power of } 2 < \ell \text{ s.t. } \ell^* | \ell.$$

For example, the first eleven values have the binary representation as given in the table:

ℓ	ℓ^*
1	1
10	10
11	1
100	100
101	1
110	10
111	1
1000	1000
1001	1
1010	10
1011	1

The intuition is that ℓ^* is giving just the bits of ℓ which are changed from $\ell - 1$.

The sequence $1, 2, \ldots, k$ is then represented by the number w whose binary representation is formed as the concatenation of the binary representations of the values $1^*, 2^*, \ldots, k^*$. For example, for $k = 11$ (eleven), the compressed encoding w will have binary representation

$$1\,10\,1\,100\,1\,10\,1\,1000\,1\,10\,1.$$

(The spaces are included only for readability.)

The intuition is that we can recover k from w by finding the entries ℓ^* in w which give bits of k: namely, the entries $\ell*$ such that no later entries has longer length. In the example, these entries are the "1000", the "10", and the "1" which are the values for ℓ^* with $\ell = 8, 10, 11$. These will be called "dominant" entries.

Lemma 1. *Let w be the number defined from k as above. Then,*

$$|w| \;=\; k + \lfloor k/2 \rfloor + \lfloor k/2^2 \rfloor + \lfloor k/2^3 \rfloor + \cdots .$$

This sum has $|k|$ many non-zero terms.

Proof. Note that there are k bits in w that are in the 1's place of some ℓ^*, for $1 \leq \ell \leq k$. There are $\lfloor k/2 \rfloor$ many such bits in the 2's place. More generally, there are $\lfloor k/2^i \rfloor$ many such bits from the 2^i's place. □

We can do even better than Lemma 1, by giving an exact formula for the length of w:

Lemma 2. *Let w be the number defined from k as above. Then,*

$$|w| \;=\; 2k - \text{Numones}(k).$$

Proof. If k is a power of two, Lemma 1 implies $|w| = 2k - 1$, so Lemma 2 follows in this case. Otherwise, express k as a sum of distinct powers of 2, $k = k_1 + k_2 + \cdots + k_i$, where $i = \text{Numones}(k)$. Clearly,

$$\lfloor k/2^j \rfloor \;=\; \lfloor k_1/2^j \rfloor + \lfloor k_2/2^j \rfloor + \cdots \lfloor k_i/2^j \rfloor.$$

Thus, we have $|w| = \sum_j (2k_i - 1)$, and Lemma 2 follows. □

This suggests to add Numones(k) many bits back into w to get the length of w exactly equal to $2k$. For this, we define $\overline{w}$ to be like w, but with an extra bit "1" inserted just before every dominant entry. For example, for k=11 again, $\overline{w}$ is equal to

$$1\,10\,1\,100\,1\,10\,1\,\underline{1}\,1000\,1\,\underline{1}\,10\,\underline{1}\,1.$$

The intuition for our Δ_0 definition of $x = 2^k$ that we define $\overline{w}$ in this way, and then note that $x = 2^k$ iff x is a power of 2 and x^2 has the same length as $2\overline{w}$. (That is, $x^2/2 \leq 2w < x^2$.) Once the graph of exponentiation is defined, it is trivial to define the length function $|x| = y$ by $2^y \leq 2x < 2 \cdot 2^y$. We can then define Numones(k) as $|\overline{w}| - |w|$.

We now give more formal details. Informally, a binary string '$100\cdots00$' is an "entry" in w if it is one of the maximal length such strings in the binary representation of w. More formally, we define IsEntryAt$(2^i, 2^{i'}, w)$ and Entry$(2^i, 2^{i'}, w) = x$ as follows. Recall that LenBit$(2^i, x)$ is the bit at position i in x; it was Δ_0-defined as $x/2^i \bmod 2$.

$$\begin{aligned}\text{IsEntryAt}(2^i, 2^{i'}, w) \quad \Leftrightarrow \quad & 2^i < 2^{i'} \wedge \text{LenBit}(2^i/2, w) = 1 \wedge \text{LenBit}(2^{i'}/2, w) = 1 \wedge \\ & (\forall 2^j \leq w)(2^i < 2^k < 2^{i'}/2 \rightarrow \text{LenBit}(2^k/2, w) = 0).\end{aligned}$$

$$\text{Entry}(2^i, 2^{i'}, w) = v \quad \Leftrightarrow \quad v = (w/2^i) \bmod (2^{i'}/2^i).$$

Here, we write $(\forall 2^j \le x)\varphi(2^j)$ as a shorthand notation for

$$(\forall z \le x)(\text{“}z \text{ is a power of 2”} \rightarrow \varphi(z)).$$

Similar conventions hold for $(\exists 2^j \le x)\varphi(2^j)$.

Note that if $\mathrm{IsEntryAt}(2^i, 2^{i'}, w)$ holds, then $\mathrm{Entry}(2^i, 2^{i'}, w)$ is a power of 2. The Entry() function picks out a power of two encoded a substring of the binary represenation of w, that is an ℓ^* value for $1 \le \ell \le k$.

A “dominant” entry in w is a entry which is longer than all later entries. In the example with $k = 11$ (eleven), there are three dominant entries: namely the binary representations “1000”, “10” and “1”. Formally,

$$\begin{aligned}\mathrm{Dominant}(2^i, 2^{i'}, w) \quad \Leftrightarrow \quad & \mathrm{IsEntryAt}(2^i, 2^{i'}, w) \wedge \\ & (\forall 2^j < 2^{j'} \le 2^i)(\mathrm{IsEntryAt}(2^j, 2^{j'}, w) \rightarrow 2^{j'}/2^j < 2^{i'}/2^i).\end{aligned}$$

The number k for which w encodes the compressed sequential counting can be extracted by the following Δ_0-defined function.

$$\begin{aligned}\mathrm{CountsTo}(w) = k \quad \Leftrightarrow \quad & (\forall 2^i \le k)[\mathrm{LenBit}(2^i, k) = 1 \leftrightarrow \\ & (\exists 2^{j'} \le w)(\mathrm{Dominant}(2^j, 2^{j'}, w) \wedge 2^i \cdot 2^j = 2^{j'}].\end{aligned}$$

The intuition is that we are determining k from w by looking at the last place where a bit of k was specified.

It is also important to be able to verify when w correctly encodes a compressed sequential counting. This is done by

$$\begin{aligned}\mathrm{IsCorrect}(w) \quad \Leftrightarrow \quad & (\forall 2^i < 2^{i'} < 2^{i''} \le w)[\mathrm{IsEntryAt}(2^i, 2^{i'}, w) \wedge \mathrm{IsEntryAt}(2^{i'}, 2^{i''}, w) \\ & \rightarrow \mathrm{CountsTo}(w/2^i) = \mathrm{CountsTo}(w/2^{i'}) + 1].\end{aligned}$$

A similar definition is used to define the correctness of $\overline{w}$, but it needs extra cases to handle the extra inserted bits after dominant entries. First, define

$$\begin{aligned}\overline{\mathrm{IsEntryAt}}(2^i, 2^{i'}, \overline{w}) \quad \Leftrightarrow \quad & \mathrm{IsEntryAt}(2^i, 2^{i'}, \overline{w}) \wedge \\ & \neg(\exists 2^j < 2^i)[\mathrm{Dominant}(2^j, 2^i, w)].\end{aligned}$$

$$\begin{aligned}\overline{\mathrm{IsCorrect}}(w) \quad \Leftrightarrow \quad & (\forall 2^i < 2^{i'} < 2^{i''} < 2^{i'''} \le w) \\ & [(\overline{\mathrm{IsEntryAt}}(2^i, 2^{i'}, w) \wedge \overline{\mathrm{IsEntryAt}}(2^{i''}, 2^{i'''}, w) \\ & \quad \wedge 2^{i''} = 2^{i'} \vee ((\mathrm{Dominant}(2^i, 2^{i'}, w) \wedge 2^{i''} = 2 \cdot 2^{i'})) \\ & \quad \rightarrow \mathrm{CountsTo}(w/2^i) = \mathrm{CountsTo}(w/2^{i''}) + 1] \\ & \wedge (\forall 2^i < 2^{i'} \le w)[\mathit{Dominant}(2^i, 2^{i'}, w) \rightarrow \mathit{LenBit}(2^{i'}, \overline{w}) = 1].\end{aligned}$$

The last clause of $\overline{\text{IsCorrect}}$ is to ensure that exactly one bit is inserted before every dominant entry.

This completes the Δ_0 definition of w and $\overline{w}$ as functions of k. Namely, there are they unique values such that IsCorrect(w) and $\overline{\text{IsCorrect}}(\overline{w})$, with CountsTo$(w) = k$ and CountsTo$(\overline{w}) = k$.

By the discussion after Lemma 2, it is now easy to Δ_0-define the graph of base 2 exponentiation, the Bit function, the LenNumones function, and the function $|x|$. The function LenNumones(x) computes the number of ones in the binary representation of $|x|$, namely LenNumones$(x) =$ Numones$(|x|)$.

Finally, we claim that all the standard "straightforward" properties of these Δ_0 relation and functions can be proved in $I\Delta_0$.

Sequence coding Sequence coding can be done using base 4 representations. We call a base four digit a *qit*[1]; namely, a qit can be a "0", "1", "2", or "3".

A sequence $\langle a_0, \ldots, a_{k-1} \rangle$ can be defined by the base 4 representation

$$3\ 1^*\ 2\ a_0\ 3\ 2^*\ 2\ a_1\ 3\ 3^*\ 2\ a_2\ 3 \cdots\ k^*\ 2\ a_{k-1}\ 3$$

The 2's and 3's mean the indicated qit 2 or 3. The values 1^*, a_0, 2^*, a_1, etc, are to be replaced by the 0/1 qits that correspond to the bits of their binary representations.

An alternate version of sequence definitions (see the handbook article) uses

$$2\ a_0\ 2\ a_1\ 2\ a_2\ 2 \cdots 2\ a_{k-1}\ 2,$$

and uses a reduction to the Numones function to count the number of qits "2" to determine the index of any entry a_i in the sequence. The qits "2" serve a commas. The Numones function is given a Δ_0 definition on pages 93-94 of [1].

References

[1] S. R. Buss, *First-order proof theory of arithmetic*, in Handbook of Proof Theory, S. R. Buss, ed., North-Holland, 1998, pp. 79–147.

[1] The only standard terms for digits in various bases seem to be: *bit*, *trit*, and *digit* for bases 2, 3, and 10; and as well as either *ban* or *hartley* for base 10, or *nat* for base e. The term "ban" was coined be A. Turing. The term *hartley* is for Ralph Hartley (1928). However, these last three terms are used for measuring information content rather than as names for digits. For names of digits in bases 2-10, I suggest: bit, trit, qit, qint, sit, sipt, ict, nint, and digit.

[2] H. Gaifman and C. Dimitracopoulos, *Fragments of Peano's arithmetic and the MRDP theorem*, in Logic and Algorithmic: An International Symposium held in honour of Ernst Specker, Monographie #30 de L'Enseignement Mathématique, 1980, pp. 187–206.

[3] P. Hájek and P. Pudlák, *Metamathematics of First-order Arithmetic*, Perspectives in Mathematical Logic, Springer-Verlag, Berlin, 1993.